Balancing Continuous Pre-Training and Instruction Fine-Tuning: Optimizing Instruction-Following in LLMs

Ishan Jindal, Chandana Badrinath, Pranjal Bharti,
Lakkidi Vinay, Sachin Dev Sharma
Samsung Research
{ishan.jindal, c.badrinath, p.bharti, l.vinay, sachin.dev}@samsung.com

Abstract

Large Language Models (LLMs) for public use require continuous pre-training to remain up-to-date with the latest data. The models also need to be fine-tuned with specific instructions to maintain their ability to follow instructions accurately. Typically, LLMs are released in two versions: the Base LLM, pre-trained on diverse data, and the instruction-refined LLM, additionally trained with specific instructions for better instruction following. The question arises as to which model should undergo continuous pre-training to maintain its instruction-following abilities while also staying current with the latest data. In this study, we delve into the intricate **relationship between continuous pre-training and instruction fine-tuning** of the LLMs and investigate the impact of continuous pre-training on the instruction following abilities of both the base and its instruction finetuned model. Further, the instruction fine-tuning process is computationally intense and requires a substantial number of hand-annotated examples for the model to learn effectively. This study aims to find the most **compute-efficient strategy** to gain up-to-date knowledge and instruction-following capabilities without requiring any instruction data and fine-tuning. We empirically prove our findings on the LLaMa 3, 3.1 and Qwen 2, 2.5 family of base and instruction models, providing a comprehensive exploration of our hypotheses across varying sizes of pre-training data corpus and different LLMs settings.

1 Introduction

Recently, autoregressive large language models (LLM) showed remarkable progress across a wide range of natural language tasks, natural language understanding, mathematical reasoning, and coding across various domains (Achiam et al., 2023; Team et al., 2024; Touvron et al., 2023; Roziere et al., 2023; Yang et al., 2024a). These LLMs are pre-trained with a causal language modeling objective to predict the next token(s) in a given sequence until it is complete, termed as *Base models*. These base models exhibit a remarkable ability to generate linguistically coherent text, however not necessarily aligning their generations with human preferences and needs (Ouyang et al., 2022). Thus, LLMs often require a fine-tuning step, *Instruction fine-tuning* to bridge the gap between the base model's fundamental objective and the practical needs of human users (Rafailov et al., 2024; Ethayarajh et al., 2024) termed as *Instruction models*.

Instruction fine-tuning is an expensive task and generally requires a significant amount of labeled data[1] depending on the type of optimization technique used[2]. This can be expensive and time-consuming to collect and annotate such a big dataset. Algorithmically, it requires training of reward model and RLHF, PPO (Ouyang et al., 2022), DPO (Rafailov et al., 2024) fine-tuning which further adds to the complexity of the task.

Parallelly, to stay abreast with the latest data, the base model needs to be either re-pre-trained on a combination of old and newly collected data (Gao et al., 2020; Tokpanov et al., 2024) or continuously pre-trained on the newly collected data (Ibrahim et al., 2024) yielding to the new base model. For example, the LLaMa 3.1 base model is pre-trained with more and high-quality data over the LLaMa 3 base model (Dubey et al., 2024). Similarly, Qwen 2.5 family base models have more knowledge and improved capabilities over Qwen 2 family models (Team, 2024).

Continuous pre-training of the LLM generally results in forgetting previously learned information, several methods have been proposed to maintain the base model performance on previously learned tasks such as Xie et al. (2023); Ibrahim et al. (2024). However, there has been no research focusing on

[1]10M hand-annotated examples were used to instruction fine-tune LLaMa 3 instruct model (AI@Meta, 2024).

[2]refer to Appendix A for more insights

the influence of continuous training on instruction models. As continuous pre-training is vital for acquiring new knowledge, and instruction tuning is necessary to learn instruction following capabilities, it is required to have both the capabilities to any instruction model. This raises a series of natural questions:

a What happens to the instruction capabilities when we continuously pre-train the instruction model to gain new knowledge?

b If lost, how to regain instruction capabilities?

c Is it necessary to add resource-extensive instruction-fine-tuning after updating the knowledge of the base model?

We approach this problem empirically by studying two different settings. In the first setting, we continuously pre-train the instruction model on a specific dataset and observe its performance on the LLM harness framework from EleutherAI (Gao et al., 2021). Whereas in another setting we continuously pre-train the base model with the same data and then instruction fine-tune the continuously pre-trained base model. Finally, we compare the instruction capabilities of instruction models from both settings. Since instruction fine-tuning is an expensive task, we discovered a simple yet efficient approach to regain the instruction capability of the continuous pre-trained base model, given that the instruction-tuned model of the original base model is available. Our main findings and the contributions of this work are as follows:

- Continuous pre-training of an instruction model results in catastrophic forgetting of the instruction capabilities and, therefore should be avoided. Section 4.1.
- Continuous pre-training base model and then instruction tuning preserve both the domain knowledge and the instruction capabilities. Section 4.4
- Instruction capabilities are portable across the same ancestor models. That is, we can extract the instruction capability by simply subtracting the weight of the base model from the weights of its instruction-tuned model. Section 4.3
- No traditional instruction tuning is required for a continuous pre-trained base model instead the instruction capabilities are ported. Section 4.2

To our knowledge, we are the first ones to systematically conduct this analysis and discover the portability of the instruction capabilities across models from the same ancestor. We empirically prove all our findings on LLaMa 3, LLaMa 3.1, Qwen2, and Qwen 2.5 families of base and instruct models. We comprehensively test our hypothesis in breadth and depth with varying sizes of pre-training data corpus across different LLMs settings in Section 3.

2 Background

In this investigation, we focus on the LLM families for which both the base model and the corresponding instruction-tuned model are publicly available. Let θ_b^{d1} be the learned parameters of the autoregressive base model on some pre-trained dataset $d1$, and θ_i^{d1v1} the corresponding parameters of instruction tuned LLM fine-tuned on some instruction dataset $v1$. Here, instruction tuning is applied on top of the base model. Given a new pre-training dataset $d2$ our objective is to obtain a $d2$ specific LLM (resulted LLM) that has the following two properties

P1 Since the $d2$ is not significantly large (as compared to $d1$) we do not want resulted LLM to forget the language understanding capabilities of the base model that it learned during the very first iteration of pre-training. Here, $d2$ itself is not sufficiently large ($<$1B tokens) to bring language understanding capabilities to any moderate-scale LLM (say 7B).

P2 Further, to align the resulted model generations with human needs, the resulted model should also have instruction following capabilities at least of the same levels as the base model.

There could be two possible settings to bring both the above properties to any LLM.

S1 Directly start with the instruction tuned LLM θ_i^{d1v1} and continuous pre-train with $d2$ dataset assuming that the resulted LLM will possess the above two properties P1, P2.

S2 First, continuously pre-train the θ_b^{d1} on $d2$ pre-training dataset via continual pre-training from Ibrahim et al. (2024) avoiding catastrophic forgetting of $d1$ learned knowledge (gain P1). Then instruction was fine-tuned with $v1$ dataset to gain instruction following capabilities (gain P2).

2.1 Resulted LLM

In this section, we explore both the above settings analyzing the anticipated pros and cons.

2.1.1 Setting 1: Continuous Pre-training of Instruction-Tuned LLM

With an assumption that the instruction capabilities of the instruction-tuned LLM will not get lost with continuous pretraining on some raw data, $d2$ is the least expensive setting to get both the new knowledge and the instruction capabilities. However, in our experiments, we could not find any evidence to validate this assumption instead observed the null hypothesis. Having said so, let's denote θ_i^{d1v1d2} are the parameters of the $d2$ continuously pre-trained instruction-tuned LLM.

2.1.2 Setting 2: Continuous Pre-training of Base LLM followed by Instruction Fine-tuning

In this setting, the base model (parameters θ_b^{d1}) is first continuously pre-trained on $d2$ dataset resulting in a new base model with parameters θ_b^{d1d2}. This updated base model has learned the new domain-specific knowledge without forgetting the initially learned knowledge (Ibrahim et al., 2024). Now, to add the instruction following capabilities this new base model needs to be instruction tuned. Let's denote θ_i^{d1d2v1} are the parameters of the resulting LLM that is instruction tuned on $d2$ continuously pre-trained LLM.

To perform instruction tuning, first, high-quality instruction-formatted data needs to be collected or constructed. Then, these formatted instances are used to fine-tune LLM in a supervised learning fashion. However, instruction tuning is an expensive time-consuming task and often poses many challenges such as clean instruction-formatted dataset (Sun et al., 2024), instruction task formatting and design (Wang et al., 2022), instruction optimization and its scalability (Xu et al., 2023), and other practical issues with instruction fine-tuning. With the availability of the original instruction fine-tuning dataset[3] some of the above-mentioned challenges might be resolved however, the practical issues such as fine-tuning stability remains.

2.2 Instruction Residuals

In this section, we describe the instruction residual approach to simply regain the instruction following capabilities [4]. We compute the instruction residual between a instruction following LLM θ_i^{d1v1} and its corresponding base model θ_b^{d1} in the parametric space as

$$\Theta_r^{v1} = \theta_i^{d1v1} - \theta_b^{d1}. \tag{1}$$

This residual computation is inspired by the parameter efficient fine-tuning of LLMs such as low-rank adaptation (LoRA (Hu et al.), QLoRA(Dettmers et al., 2024), DoRA (Liu et al., 2024) etc). In these techniques instead of fine-tuning a large weight matrix W for a given layer, a low-rank ΔW matrix is learned, which contains the new information to be integrated with the original model that is $W_{updated} = W + \Delta W$. These techniques add new information/capabilities to the original model often with fewer parameters defined by *rank* of ΔW. With the full ΔW rank, it is similar to fine-tuning the whole model end-to-end (Hu et al.).

Inspired by this idea of weight addition to learning a new capability, we first extract the instruction capability by subtracting the base LLM weights from its corresponding instruction-tuned LLM weights as in 1, termed as *instruction residuals*, and add this instruction residual to the continuously pre-trained base LLM on new skill $d2$ that is

$$\theta_i^{d1d2v1} = \theta_b^{d1d2} \oplus \Theta_r^{v1}, \tag{2}$$

where $\oplus$ represents element-wise addition. These tensor addition and subtraction to regain the instruction capabilities do not incur heavy computation costs making the instruction-tuned LLM readily available once the new knowledge is learned by the base LLM. One major limitation of this work is that if the base LLM and its corresponding instruction-tuned LLM are not available then the instruction residuals from 1 won't be available and hence requires a full cycle of instruction fine-tuning to regain this ability.

3 Experiments

3.1 Datasets

3.1.1 Pre-traininig Dataset

We need this pre-training dataset to test the impact on instruction capabilities of the continuously pre-trained model. We want the new pre-training data

[3]Instruction fine-tuning datasets used to tune LLaMa, Qwen, and other SoTA LLMs are not shared in public

[4]Instruction following capabilities and instruction capabilities are used interchangeably in this work

Category	Sub Category	Benchmark
Instruction following	Language understanding	IFEval
		MMLU MMLU-Pro
	Math and logic	GSM8K
Reasoning and problem solving	Commonsense	Winogrande Hellaswag
	Factual knowledge	ARC_easy
	Physical reasoning	Piqa
Truthfulness		Truthfulqa_mc2

Table 1: Evaluation dataset categorization.

such that none of the base models and the corresponding instruct model have seen that data previously. Since the data contamination is a serious concern as noted in Jiang et al. (2024), the existing pre-training datasets may not be the right choice to continuously pre-train the model. Therefore, we manually scraped around 2M articles using a static news crawler FUNDUS[5] (Dallabetta et al., 2024). We choose the news articles that are new to LLaMa 3.1 models that is we choose the articles published in the date range from December 2023 to September 2024 from all existing publishers in FUNDUS. The average length of the articles is 650 LLaMa tokens with 6981 max tokens and 156 min tokens. These articles are then packed with a sequence length of 4096 (LLaMa maximum sequence length is 8K but we choose 4K to efficiently utilize the existing GPU vRAM), and similar to Kosec et al. (2021) we use attention masks for each article to avoid cross article contamination.

3.1.2 Evaluation Dataset

In this section, we describe the test dataset used to evaluate our hypothesis. To perform a comprehensive evaluation and to maintain reproducibility we use the evaluation harness framework from EleutherAI (Gao et al., 2021). We particularly target to evaluate the following capabilities:

Instruction following

IFEval This dataset was introduced mainly to focus on natural language instruction following capabilities of LLMs (Zhou et al., 2023). It contains 25 types of verifiable instructions such as *write in more than 400 words*, *mention the keyword of AI at least 3 times* with 500 prompts. This evaluation is performed on 4 metrics: (1) Prompt-level strict-accuracy (PLS-acc): The percentage of prompts that all verifiable instructions in each prompt are followed, (2) Inst-level strict-accuracy (ILS-acc): The percentage of verifiable instructions that are followed, (3) Prompt-level loose-accuracy (PLL-acc): Prompt-level accuracy computed with the loose criterion, and (4) Inst-level loose-accuracy (ILL-acc): Instruction-level accuracy computed with a loose criterion.

MMLU mainly focuses on extensive world knowledge across 57 subjects which includes all major domains like math, computer science, medicine, philosophy, and law (Hendrycks et al., 2020). This dataset contains a total of 15908 development and test questions with 4 possible answers each.

MMLU-Pro is introduced to further increase the complexity of the MMLU benchmark since the existing LLMs are excelled at MMLU (Wang et al., 2024). This dataset was curated by eliminating some trivial and noisy questions from MMLU and by introducing reasoning-focused questions to MMLU which has mostly knowledge-driven questions.

GSM8K dataset consists of 8.5K high-quality linguistically diverse grade school math problems (Cobbe et al., 2021). These problems take between 2 and 8 steps to solve, and solutions primarily involve performing a sequence of elementary calculations using basic arithmetic operations ($+, -, \times, \div$) to reach the final answer. **Reasoning and problem-solving**

Winogrande is a large scale 44k commonsense reasoning dataset. It mainly tests a model's ability to resolve ambiguous pronouns based on contextual understanding (Sakaguchi et al., 2021).

Hellaswag is designed to benchmark commonsense reasoning in AI models (Zellers et al., 2019). It contains 10,000 multiple-choice questions for validation and testing. The dataset focuses on predicting the most plausible continuation of a given scenario.

ARC_easy dataset consists of a collection of 7787 natural science questions (Clark et al., 2018). The dataset contains only natural, grade-school science questions. ARC questions appeal to both different styles of knowledge and different styles of reasoning.

Piqa evaluates the model on the physical commonsense questions without experiencing the physical world (Bisk et al., 2020). Each instruction has a goal to reach in the physical world, given the description of the environment if required, and 2

[5]https://github.com/flairNLP/fundus

Benchmark	Metric	L3b				L3i				L3b + 3Lr		
# of new tokens →		org.	+100M	+500M	+1B	org.	+100M	+500M	+1B	100M	500M	1B
IFEval	ILL_acc	19.06	17.75	19.30	20.14	53.36	45.68	45.32	41.01	57.67	56.47	57.79
	ILS_acc	17.87	16.31	17.87	17.87	47.84	40.41	38.61	35.25	51.68	51.44	51.68
	PLL_acc	09.98	09.61	10.54	11.09	41.77	34.20	33.64	28.10	44.18	42.88	44.36
	PLS_acc	09.06	08.69	09.61	09.80	35.30	29.21	26.80	22.55	37.52	36.78	37.52
MMLU	acc	62.14	63.76	63.77	63.62	63.83	66.04	65.38	65.52	67.69	67.16	67.51
MMLU-Pro	EM	34.51	34.92	34.70	35.49	39.70	36.39	35.76	35.52	40.72	40.84	40.27
GSM8K	EM	49.58	48.14	47.54	47.08	75.06	69.22	68.08	68.01	74.83	73.92	73.16
	strict-EM	49.20	34.04	36.62	39.04	74.98	65.35	59.74	55.42	46.93	55.27	49.51
Sub-Average		31.43	29.15	29.99	30.52	**53.98**	48.31	46.67	43.92	52.65	<u>53.10</u>	52.73
Winogrande	acc	73.16	72.14	71.59	71.27	71.74	72.22	71.74	71.74	71.43	72.06	71.19
Hellaswag	acc	60.12	60.17	60.27	60.23	57.70	58.69	58.73	59.00	59.30	59.16	59.39
	acc_n	79.22	78.62	78.20	78.36	75.76	77.84	77.68	77.80	79.01	79.14	79.00
ARC_easy	acc	80.39	81.14	80.60	80.60	81.52	79.71	79.63	79.46	82.20	82.32	82.03
	acc_n	77.78	80.30	79.67	79.38	79.63	77.53	77.65	77.44	79.00	79.25	79.08
Piqa	acc	79.54	80.09	80.30	80.20	78.56	79.87	79.49	79.49	80.25	80.20	80.41
	acc_n	80.74	81.56	81.34	80.96	78.62	80.41	79.98	79.98	80.63	80.74	80.90
Sub-Average		75.85	**76.29**	76.00	75.86	74.79	75.18	74.99	74.99	75.97	<u>76.12</u>	76.00
T_mc2	acc	43.94	47.64	47.29	47.41	51.67	51.80	51.55	51.68	<u>56.13</u>	55.73	**56.17**
Average		51.64	50.93	51.20	51.41	62.94	60.28	59.36	58.00	63.07	63.34	63.12

Table 2: Impact of continual pretraining on LLaMa 3 Base (L3b) and the LLaMa 3 instruction tuned (L3i) models w.r.t varying number of new tokens. Also, depicts the usefulness of the instruction residual technique to regain instructional capabilities.

options (solutions) to reach the goal. We report both the accuracy and the normalized accuracy on this dataset.

Truthfulness measure the truthfulness of a language model in answering questions (Lin et al., 2021). This dataset consists of 817 questions across 38 categories and captures human misconceptions, false beliefs, conspiracies, and awareness between real-world knowledge and fictional knowledge across 38 domains including health, law, finance, and politics. Because of space constraints, we abbreviate this dataset as *T_mc2*.

We choose these datasets as these are commonly evaluated for most of the newly released LLMs (Touvron et al., 2023; Yang et al., 2024a). Table 1 summarizes all the evaluation datasets used in this work for each category. We used the latest versions of these datasets available on EleutherAI[6] as of writing this work. Only MMLU, MMLU-Pro, and GSM8K are evaluated with 5-shot, rest of the datasets are evaluated on zero-shot.

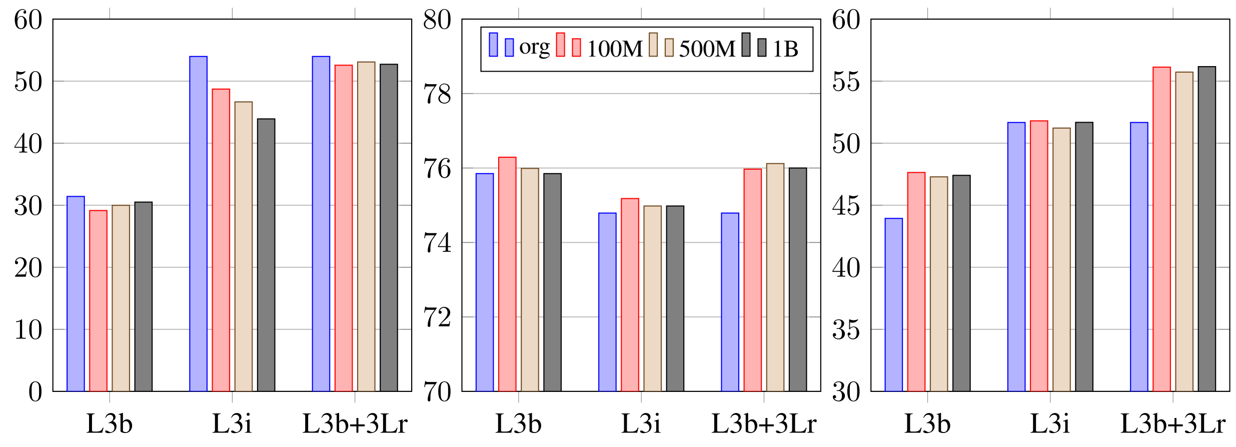
[6]`https://github.com/EleutherAI/lm-evaluation-harness`

3.2 Language Model Architectures

We used two distinct families of language models LLaMa (Dubey et al., 2024) and Qwen (Yang et al., 2024a). Specifically, we target the LLaMa 3, 3.1 family of models, and the Qwen 2, 2.5 family of models. Both LLaMa 3 and 3.1 are available in 8B, 70B parameters size with the exception that the 3.1 family also has a 405B parameters model. For all our LLaMa experiments we focused only on the 8B models because of the resource constraints. Similarly, both Qwen 2 and 2.5 come in 0.5B, 1.5B, and 7B parameter models with the exception that the 2.5 family also has 3B, 14B, 32B, and 72B parameter models. For all Qwen experiments, we choose 0.5B, 1.5B, and 7B models. Further, by design, we are required to choose a family of models for which both the base and the instruction-tuned variants of the same size exist.

4 Results and Analysis

In this section, we will examine the influence of continuous pre-training on the instructional capabilities of both the base model and its corresponding instruct models. Specifically, we will determine **if an additional round of fine-tuning is neces-**



(a) Instruction Following (b) Reasoning (c) Truthfulness

Figure 1: Impact of continual pre-training on instruction following capability of LLaMa base and instruct models compared against residual technique.

sary after updating the Large Language Model (LLM) knowledge. Subsequently, we will assess the efficacy of a proposed instruction residuals technique, which is simple yet effective, to **restore any lost instructional capabilities** without explicit instruction fine-tuning of the updated LLM.

4.1 Impact of Continual Pretraining

In this section, we delve into the effects of continual pre-training on the LLaMa 3 8B models, specifically focusing on both the LLaMa 3 base (L3b) and the LLaMa 3 instruction-tuned (L3i) models. The pre-training process was carried out over a diverse set of new tokens, with quantities of 100M, 500M, and 1B, to comprehensively investigate the impact of the new token size on instructional capabilities (detailed in Section 3.1.1).

Figure 1 provides the summary of our findings. Notably, as the instruction model (L3i) encounters an increasing number of new tokens, its instructional capabilities deteriorate in Figure 1a. For instance, with 100M new tokens, we observed a maximum drop of 5.7 points on the instruction following dataset and an average drop of 2.7 points across all tasks. This trend intensifies as we add more tokens to the continuous pre-training datasets. That is with 1B new tokens, resulting in a maximum drop of 10 points on the instruction following dataset and an average drop of 3 points overall. Hence, confirming that **the continual pre-training of instruction-tuned models with a large volume of new tokens leads to a significant loss of its instructional capabilities**.

In contrast, the base model (L3b) in Figure 1 demonstrates minimal quality degradation with the number of new tokens. While it is anticipated that base models to lack instructional capabilities, we do not observe significant catastrophic forgetting of any such capabilities with continual pre-training. Hence, **base models appear to be less susceptible to this effect, maintaining relatively stable performance despite an increase in new tokens**.

4.2 Restore Instruction Capabilities

As described in Section 2 we restore the instructional capabilities by incorporating instructional residuals Θ_r^{v1} into a continuously pre-trained base model, where Θ_r^{v1} is calculated as the difference between the weights of the instruction tuned model θ_i^{d1v1} and the weights of the base model θ_b^{d1}.

The results, as demonstrated in Table 2 (last column block), reveal a significant improvement in model performance when the instruction residual technique is employed. Specifically, the residual-adjusted models (L3b + 3Lr) outperform the L3i models by an average of 4 absolute points across all tasks when pre-trained on 500M new tokens. This improvement is not only consistent but also increases with the number of new tokens, reaching a significant improvement of 5 absolute points when pre-trained on 1B new tokens.

Based on these findings, we assert that an **additional round of instruction fine-tuning is necessary after updating the instruction-tuned model's knowledge**. Furthermore, we posit that the **instructional residuals not only restore the instructional capabilities of the continuously pre-trained model** but also enhance these abilities on

LLM (Params) →		**LLaMa 3 (8B)**			**LLaMa 3.1 (8B)**			**Qwen 2 (1.5B)**			**Qwen 2.5 (1.5B)**		
Benchmark	Metric	L3b	L3i	+3.1Lr	L3.1b	L3.1i	+3Lr	Q2b	Q2i	+2.5Qr	Q2.5b	Q2.5i	+2Qr
IFEval	ILL_acc	19.06	53.36	**57.19**	15.59	**54.80**	48.32	25.84	**29.02**	27.22	28.15	**39.09**	26.50
	ILS_acc	17.87	47.84	**51.92**	14.63	**49.88**	43.76	23.62	**26.74**	23.86	25.72	**35.21**	25.06
	PLL_acc	09.98	41.77	**43.81**	07.76	**41.59**	34.38	16.08	**17.93**	17.19	20.01	**25.67**	20.33
	PLS_acc	09.06	35.30	**37.52**	07.02	**35.49**	29.39	14.05	**15.71**	14.23	17.65	**22.04**	18.85
MMLU	acc	62.14	63.83	**66.02**	63.40	**68.03**	64.89	55.10	**55.80**	55.16	59.75	**59.75**	59.75
MMLU-Pro	EM	34.51	**39.70**	38.53	35.32	**40.97**	40.40	21.21	**21.70**	21.10	27.13	**29.89**	27.00
GSM8K	EM	49.58	75.06	**75.13**	50.11	**76.19**	73.84	54.51	**58.30**	57.01	57.39	56.86	**58.76**
	strict-EM	49.20	74.98	**74.98**	49.81	**75.36**	73.84	54.44	**57.39**	56.79	57.16	53.37	**58.30**
Sub-Average		31.43	53.98	**55.64**	30.46	**55.29**	51.10	33.11	**35.32**	34.07	36.62	**40.24**	36.82
Winogrande	acc	**73.16**	71.74	72.77	**73.64**	73.40	73.01	**66.38**	65.19	65.82	63.22	**65.19**	63.22
Hellaswag	acc	**60.12**	57.70	59.00	**60.02**	59.10	58.25	48.61	**49.30**	48.58	50.16	**50.94**	50.08
	acc_n	**79.22**	75.76	78.88	78.90	**79.19**	76.68	65.43	**66.07**	65.42	67.81	**68.34**	67.76
ARC_easy	acc	80.39	81.52	**81.57**	81.40	81.94	**82.91**	66.25	**69.99**	66.37	75.42	**76.56**	75.67
	acc_n	77.78	**79.63**	78.83	81.14	79.76	**81.86**	60.86	**66.54**	60.14	71.84	**76.26**	72.26
Piqa	acc	**79.54**	78.56	79.00	**80.09**	80.03	79.16	75.41	**76.17**	75.95	75.68	**76.39**	75.84
	acc_n	**80.74**	78.62	80.14	81.07	**81.12**	79.22	75.41	**75.90**	75.79	76.06	**76.12**	75.79
Sub-Average		**75.85**	74.79	75.74	**76.61**	76.36	75.87	65.48	**67.02**	65.48	68.60	**69.97**	68.66
T_mc2	acc	43.94	51.67	**52.73**	45.17	**53.92**	52.20	45.95	43.36	**45.95**	**46.64**	46.65	46.45
Average		51.64	62.94	**64.25**	51.57	**64.42**	62.01	48.07	**49.69**	48.54	51.24	**53.65**	51.35

Table 3: Instruction portability quality comparison with LLaMa 3 and 3.1 8B LLMs. Where the **BOLD** represents the best quality and the underline shows the second-best score for that column block.

numerous tasks.

4.3 Instruction Portability across LLM Families

We summarize the impact of instruction residuals across different LLM families with varying model sizes in Table 3, where *{x}b* represents the base model, *{x}i* represents the corresponding instruction tuned model and *+{x}r* represents the instruction residual adjusted continuously pre-trained base model. For example, *+3.1Lr* means instruction residuals of LLaMa 3.1 family $\theta^{d1d2v1}_{L3.1i} - \theta^{d1d2}_{L3.1b}$ are added to the LLaMa 3 base model θ^{d1}_{L3b}. Similarly, *+2.5Qr* means instruction residuals of Qwen 2.5 family $\theta^{d1d2v1}_{Q2.5i} - \theta^{d1d2}_{Q2.5b}$ are added to the Qwen 2 base model θ^{d1}_{Q2b}.

For the LLaMa family, we observe that the instruction residuals always improve the instruction-following capabilities of the base model. For both LLaMa 3 and 3.1 we observe a consistent gain over the base model for all the datasets. As noted in the LLaMa 3.1 technical report, it possesses high-quality instruction-following capabilities, Therefore, LLaMa 3.1 instruction residuals are expected to carry better instruction-following capabilities than LLaMa 3's instruction residuals. As expected, instruction residuals of LLaMa 3.1 (*3.1Lr*), when merged with the LLaMa 3 base model (*L3b*), improve its instruction capabilities (64.25 Vs 51.64) better than its own instruction fine-tuned model (*L3i*) by 1 absolute point. On the other hand, since LLaMa 3 is inferior in quality to LLaMa 3.1, its instruction residuals (*3Lr*) also possess low-quality instruction-following capabilities than instruction residuals of LLaMa 3.1 (*3.1Lr*). Therefore, as expected, instruction residuals of LLaMa 3 (*3Lr*) when merged with the LLaMa 3.1 base model (*L3.1b*) improves its instruction capabilities (62.01 Vs 51.57). However, performs lower than its original instruction fine-tuned model (*L3.1i*). One observation that stands out in this experiment is that the model with instruction residuals performs always better than the corresponding base models. Hence prove that **the instruction capabilities are portable across models of the same family LLMs**.

4.4 Instruction Residual Applicability to Derived LLMs

In this section, we investigate the applicability of instruction residuals on the LLMs derived from the same ancestor. Specifically, we choose an

Benchmark	Metric	DocChat	+3Lr	+3.1Lr
IFEval	ILL_acc	38.25	49.64	56.71
	ILS_acc	34.65	46.04	53.12
	PLL_acc	24.95	37.34	44.36
	PLS_acc	20.89	32.90	39.74
MMLU	acc	62.96	62.31	65.35
MMLU-Pro	EM	36.36	39.66	39.36
GSM8K	EM	57.09	78.54	74.53
	strict-EM	56.94	77.48	69.90
Winogrande	acc	74.27	71.27	73.32
Hellaswag	acc	61.68	57.57	59.51
	acc_n	80.36	75.79	78.39
ARC_easy	acc	82.11	81.02	80.22
	acc_n	81.52	79.00	77.99
Piqa	acc	80.47	78.13	78.78
	acc_n	81.61	77.97	78.62
T_mc2	acc	45.35	49.54	50.82
Average		57.47	<u>62.14</u>	**63.80**

Table 4: Applicability of instruction residual approach on publicly available *Llama3-DocChat-1.0-8B* (DocChat) LLM that was built on top of LLaMa 3 base model. Here, +3Lr and +3.1Lr are the instruction residuals from LLaMa 3 and LLaMa 3.1, respectively integrated to the original DocChat LLM.

LLM that is either continuously pre-trained or instruction-tuned on the LLaMa 3 base model. We find a plethora of LLMs derived from LLaMa 3 on HuggingFace [7]. Among existing, we chose to experiment with *cerebras/Llama3-DocChat-1.0-8B* as this model makes the LLaMa 3 base model specialized for a particular task. Specifically, this LLM adds a new document-based QA skill to the LLaMa 3 base model via instruction fine-tuning the base model with the ChatQA dataset. Table 4 summarizes the impact of instruction residuals on the instruction following capabilities of this LLM. It is evident that the instruction residual significantly improves the instruction-following capabilities of the DocChat LLM both with LLaMa 3 and 3.1 residuals. The impact is more pronounced with the LLaMa 3.1 residuals reaching a gain of 6 absolute points in quality. Hence, it is evident that the **instruction residuals are portable across same ancestor models**.

5 Related Work

Continual Learning is a well-established technique for updating existing machine learning models with the latest information and trends, allowing language models to adapt to new data while preserving the knowledge acquired during prior training (Caccia et al., 2020; Le Scao et al., 2023; Ibrahim et al., 2024). Several studies have applied continuous pre-training in large language models (LLMs) to acquire new skills, domains, and languages, and perform various tasks, as demonstrated in works such as Yadav et al. (2023); Ma et al. (2023); Yang et al. (2024c); Gogoulou et al. (2023).

Yang et al. (2024c) adopt a strategy of continuous pre-training followed by instruction fine-tuning on domain-specific data to learn new domains. However, the reasons behind why this approach is the most effective for acquiring both new knowledge and instruction-following capabilities in large language models (LLMs) have not been thoroughly explored.

Model Merging: Recent studies have demonstrated that specialized fine-tuned models can be merged to combine capabilities and generalize to new skills (Yu et al.; Yang et al., 2024b). Several techniques have been explored for merging the abilities of two or more models, including Task Arithmetic (Ilharco et al., 2022), TIES (Yadav et al., 2024), and Model Breadcrumbs (Davari and Belilovsky, 2023). Following Ilharco et al. (2022), we employ Task Arithmetic in our work to extract the instruction residual. Although different model merging techniques could affect the transfer of instruction-following capabilities, understanding the specific impact of these techniques is beyond the scope of this study.

6 Conclusion

In conclusion, this study delves into the effects of continuous pre-training on base and instruction-tuned large language models (LLMs) and their instruction capabilities. The findings suggest that while continuous pre-training of instruction models may lead to catastrophic forgetting of instruction capabilities, a more efficient approach is to continuously pre-train the base model with new data, followed by instruction tuning. This method preserves both domain knowledge and instruction capabilities. Interestingly, the study also reveals that instruction capabilities are transferable across models from the same ancestor, eliminating the need for additional instruction tuning for a continuously pre-trained base model. We empirically demonstrated this analysis on the LLaMa 3 and LLaMa 3.1 family of base and instruction models.

[7]https://huggingface.co/models?search=llama3

Limitations

While our hypothesis is validated for models with 8 billion parameters, we observe a noticeable variation in performance when applied to smaller models, particularly those with around 1.5 billion parameters. Furthermore, the scalability of our proposed strategy for models smaller than 1.5 billion parameters remains uncertain. This presents an intriguing avenue for future research, where further exploration could investigate whether modifications or optimizations are needed to maintain the same level of effectiveness for these smaller models.

A critical challenge that emerges with the instruction residual method is the reliance on the availability of both the base language model and its instruction fine-tuned counterpart. The approach fundamentally depends on the residual differences between these two models to function effectively. In the absence of either the base model or the fine-tuned model, the instruction residual method cannot be employed. This limitation highlights a bottleneck in the methodology, especially when resources or computational constraints prevent the simultaneous availability of both models. Future work could explore potential ways to mitigate this dependency, perhaps by developing alternative techniques that either reduce the need for dual-model structures or enhance the portability of instruction-based fine-tuning across a wider range of model sizes.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

AI@Meta. 2024. Llama 3 model card.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Caccia, Issam Laradji, Irina Rish, Alexandre Lacoste, David Vazquez, et al. 2020. Online fast adaptation and knowledge accumulation: a new approach to continual learning. *arXiv preprint arXiv:2003.05856*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Max Dallabetta, Conrad Dobberstein, Adrian Breiding, and Alan Akbik. 2024. Fundus: A simple-to-use news scraper optimized for high quality extractions. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 305–314, Bangkok, Thailand. Association for Computational Linguistics.

MohammadReza Davari and Eugene Belilovsky. 2023. Model breadcrumbs: Scaling multi-task model merging with sparse masks. *arXiv preprint arXiv:2312.06795*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. 2021. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, 10:8–9.

Evangelia Gogoulou, Timothée Lesort, Magnus Boman, and Joakim Nivre. 2023. A study of continual learning under language shift. *arXiv preprint arXiv:2311.01200*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L Richter, Quentin Anthony, Timothée Lesort, Eugene Belilovsky, and Irina Rish. 2024. Simple and scalable strategies to continually pre-train large language models. *arXiv preprint arXiv:2403.08763*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

Minhao Jiang, Ken Liu, Ming Zhong, Rylan Schaeffer, Siru Ouyang, Jiawei Han, and Sanmi Koyejo. 2024. Does data contamination make a difference? insights from intentionally contaminating pre-training data for language models. In *ICLR 2024 Workshop on Navigating and Addressing Data Problems for Foundation Models*.

Matej Kosec, Sheng Fu, and Mario Michael Krell. 2021. Packing: Towards 2x nlp bert acceleration.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

Shirong Ma, Shen Huang, Shulin Huang, Xiaobin Wang, Yangning Li, Hai-Tao Zheng, Pengjun Xie, Fei Huang, and Yong Jiang. 2023. Ecomgpt-ct: Continual pre-training of e-commerce large language models with semi-structured data. *arXiv preprint arXiv:2312.15696*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. 2024. Principle-driven self-alignment of language models from scratch with minimal human supervision. *Advances in Neural Information Processing Systems*, 36.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Yury Tokpanov, Beren Millidge, Paolo Glorioso, Jonathan Pilault, Adam Ibrahim, James Whittington, and Quentin Anthony. 2024. Zyda: A 1.3 t dataset for open language modeling. *arXiv preprint arXiv:2406.01981*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.

Yong Xie, Karan Aggarwal, and Aitzaz Ahmad. 2023. Efficient continual pre-training for building domain specific large language models. *arXiv preprint arXiv:2311.08545*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Prateek Yadav, Qing Sun, Hantian Ding, Xiaopeng Li, Dejiao Zhang, Ming Tan, Xiaofei Ma, Parminder Bhatia, Ramesh Nallapati, Murali Krishna Ramanathan, et al. 2023. Exploring continual learning for code generation models. *arXiv preprint arXiv:2307.02435*.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2024. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 2024b. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*.

Xianjun Yang, Junfeng Gao, Wenxin Xue, and Erik Alexandersson. 2024c. Pllama: An open-source large language model for plant science. *arXiv preprint arXiv:2401.01600*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Overview of Experimental Hardware

A.1 GPU Specifications

- GPU: NVIDIA A100 40GB SXM
- GPU Memory: 40GB
- FP16/BF16 Tensor Core: 312 TeraFLOPs
- TF32: 156 TeraFLOPs

A.2 FLOPs Requirements

Instruction Fine-tuning

- Number of Parameters, N: 8B.
- Number of tokens, $tokens$

$$\begin{aligned} tokens &: 25M samples \\ &\approx 25M \times 8192 \\ &= 204,800\text{M tokens} \end{aligned}$$

- Number of Epochs, E: Fine-tuning generally requires fewer epochs; often 3 to 10 epochs are sufficient.

Continued Pre-training The below calculations assume continuous pre-training with 100M Tokens.

- Number of Parameters, N: 8B.
- Number of tokens, $tokens$: 100M tokens
- Sequence Length, S: 4096
- Number of Epochs, E: 5

Estimate FLOPs per Tokens

The FLOPs per training step depend on the number of operations performed per token per layer. Assuming each parameter needs about 6 floating-point operations (forward and backward):

$$\text{FLOPs/token/parameter} \approx 6$$

Given the structure of transformers with multiple layers and self-attention, let's simplify and assume each token requires 6 operations per parameter across all layers:

$$\text{FLOPs/token} = 6 \times N$$

A.3 A Comparison

Here, we perform the comparison between LLaMa8B Instruction Tuning and 100M continous pre-training (CP) in terms of numbers of FLOPs.

$$\begin{aligned} ratio &= \frac{\text{Instruct}(6 \times 8 \times 10^9 \times tokens \times E)}{\text{CP}(6 \times 8 \times 10^9 \times tokens \times E)} \\ &= \frac{6 \times 10^9 \times 204800 Million \times 5}{6 \times 10^9 \times 100 Million \times 5} \\ &\approx 2048 \end{aligned}$$

This calculation provides a rough estimate of the FLOPs required to continue pre-training the model on 100M tokens across 5 epoch and instruction fine-tuning FLOPs, estimates numbers from Llama3 Paper.

A.4 MMLU Performance vs Compute Cost

we demonstrate how our model maintained a good enough performance on the MMLU benchmark while optimizing for low computational costs. The results reflect efficient usage of available compute resources, particularly by leveraging hardware such as the NVIDIA A100.

A.4.1 Comparison of MMLU Scores and Compute Costs

Our approach achieved high accuracy in various tasks under the MMLU benchmark, matching the performance of more compute-intensive models. Despite this, we successfully reduced the total compute cost by optimizing training and fine-tuning processes, as shown in Figure 2.

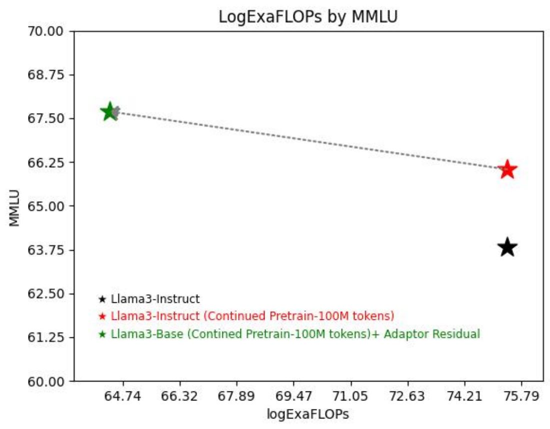


Figure 2: Comparison of MMLU performance and compute costs across different models. Our model (marked in green) balances compute efficiency while maintaining competitive performance.